UNIVERSITÀ DI PARMA
Dipartimento di Ingegneria e Architettura

Public Key (asymmetric) Cryptography

Luca Veltri

(mail.to: luca.veltri@unipr.it)

Course of Cybersecurity, 2022/2023

http://netsec.unipr.it/veltri

Public-Key Cryptography

- Also referred to as asymmetric cryptography or two-key cryptography
- Probably most significant advance in the 3000 year history of cryptography
  - **public invention due to Whitfield Diffie & Martin Hellman in 1975**
    - at least that's the first published record
    - known earlier in classified community (e.g. NSA?)
- Is asymmetric because
  - **who encrypts messages or verify signatures cannot decrypt messages or create signatures**
  - **more in general, operation performed by two parties use different key values**

Public-Key Cryptography (cont.)

- Public-Key cryptography uses clever application of number theoretic concepts and mathematical functions rather than permutations and substitutions
- Makes use of "trapdoor functions"
  - **a trapdoor function is a function *f(x)* that is fast to be computed, while its inverse is hard to be computed, unless a secret information *t* (the trapdoor) is known**
    - when *t* is known, it is also easy to compute *x* from *f(x)*
  - **example of trapdoor function:**
    - prime number product
      - it is easy to find the product of two big prime numbers
      - it is hard to factorize a composite (two factors) big integer
      - it is easy to factorize a composite (two factors) big integer when one of the two factor is known

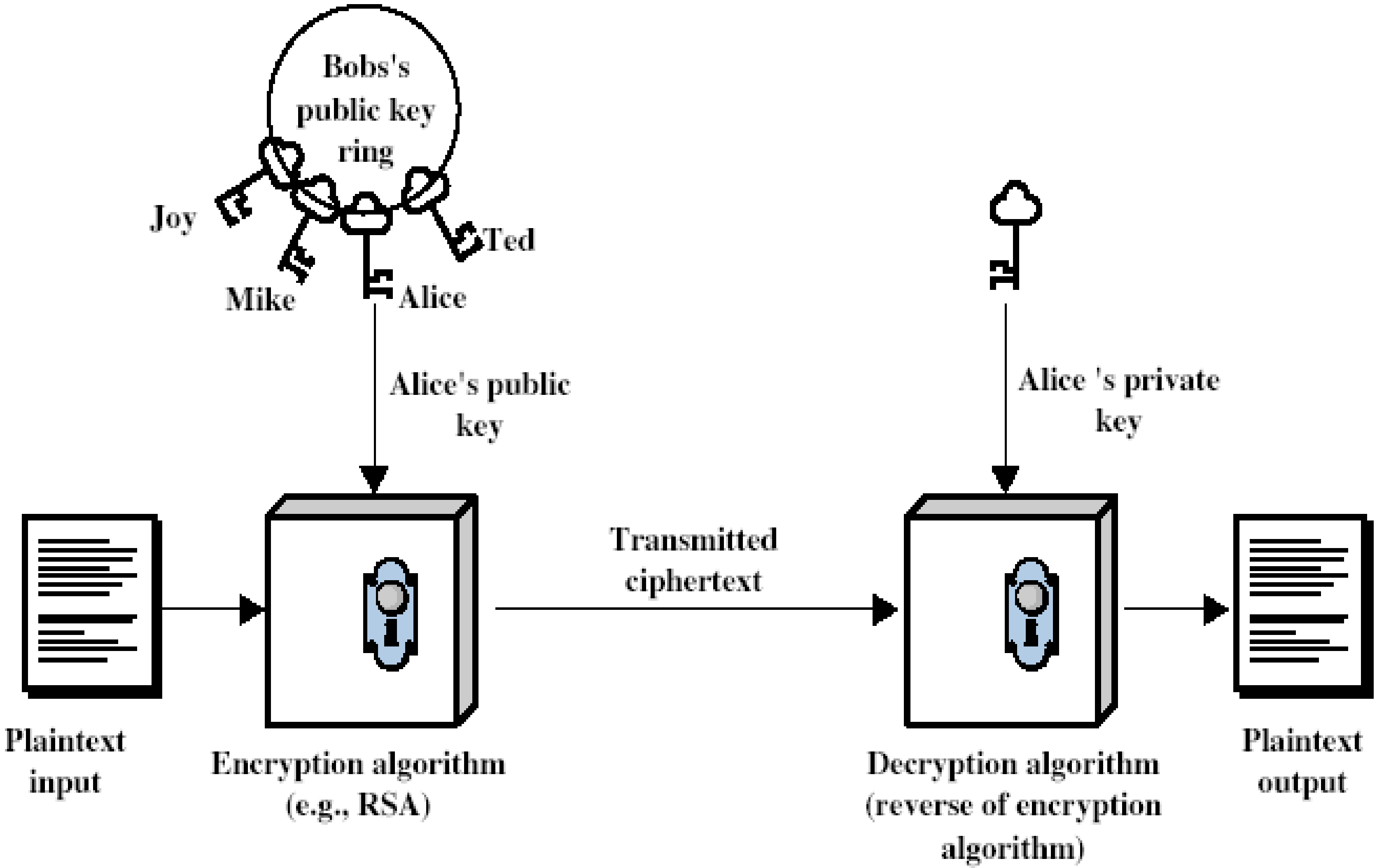
Public-Key vs. Secret Cryptography

- All secret key algorithms do the same thing
  - **they take a block and encrypt it in a reversible way**
- All hash (and MAC) algorithms do the same thing
  - **they take a message and perform an irreversible transformation**
- Instead, public key algorithms look very different
  - **in how they perform their function**
  - **in what functions they perform**
- They have in common: a private and a public quantities associated with a principal

Public-Key vs. Secret Cryptography (cont.)

- Pub-key vs. secret key management
  - **With symmetric/secret-key cryptography**
    - you need a secure method of telling your partner the key
    - you need a separate key for everyone you might communicate with
  - **Instead, with public-key cryptography, keys do not have to be secretly shared**
  - **Public-key cryptography often uses two keys:**
    - a public-key, which may be known by anybody, and can be used to encrypt messages, or verify signatures
    - a private-key, known only to the recipient, used to decrypt messages, or sign (create) signatures
    - it is computationally easy to en/decrypt messages when key is known
    - it is computationally infeasible to find decryption key knowing only encryption key (and vice-versa)
  - **Some asymmetric algorithms don’t use keys at all!**

Public-Key Cryptography



Public-Key vs. Secret Cryptography (cont.)

- Public key cryptography can do anything secret key cryptography can do, but..
  - **simpler key initialization**
  - **more accurate key-to-entity association**
    - private key is known only by the owner
  - **slower execution**
    - orders of magnitude slower than the best known secret key cryptographic algorithms
- They are usually used only for things secret key cryptography can't do (or can't do in a suitable way)
- Complements rather than replaces secret key crypto
  - **often it is mixed with secret key technology**
  - **e.g. public key cryptography might be used in the beginning of communication for authentication and to establish a temporary shared secret key used to encrypt the conversation**

Why Public-Key Cryptography?

- Can be used to:
  - **key distribution – secure communications without having to trust a KDC with your key (key exchange)**
  - **digital signatures –verify a message is come intact from the claimed sender (authentication)**
  - **encryption/decryption - secrecy of the communication (confidentiality)**
- Note:
  - **public-key cryptography simplifies but not eliminates the problem of key management**
  - **some algorithms are suitable for all uses, others are specific**
- Example of public key algorithms:
  - **RSA, which does encryption and digital signature**
  - **DSS, which do digital signature but not encryption**
  - **Diffie-Hellman, which allows establishment of a shared secret**
  - **Fiat-Shamir identification scheme, which only do authentication**

Security of Public Key Schemes

- Security of public-key algorithms still relies on key size (as for secret-key algorithms)
- Like private key schemes brute force exhaustive search attack is always theoretically possible
  - **But keys used are much larger (>512bits)**
- A crucial feature is that the private key is difficult to determine from the public key
  - **security relies on a large enough difference in difficulty between easy (en/decrypt) and hard (cryptanalyse) problems**
  - **often the hard problem is known, its just made too hard to do in practise**
    - requires the use of very large numbers
    - hence is slow compared to private key schemes

RSA Algorithm

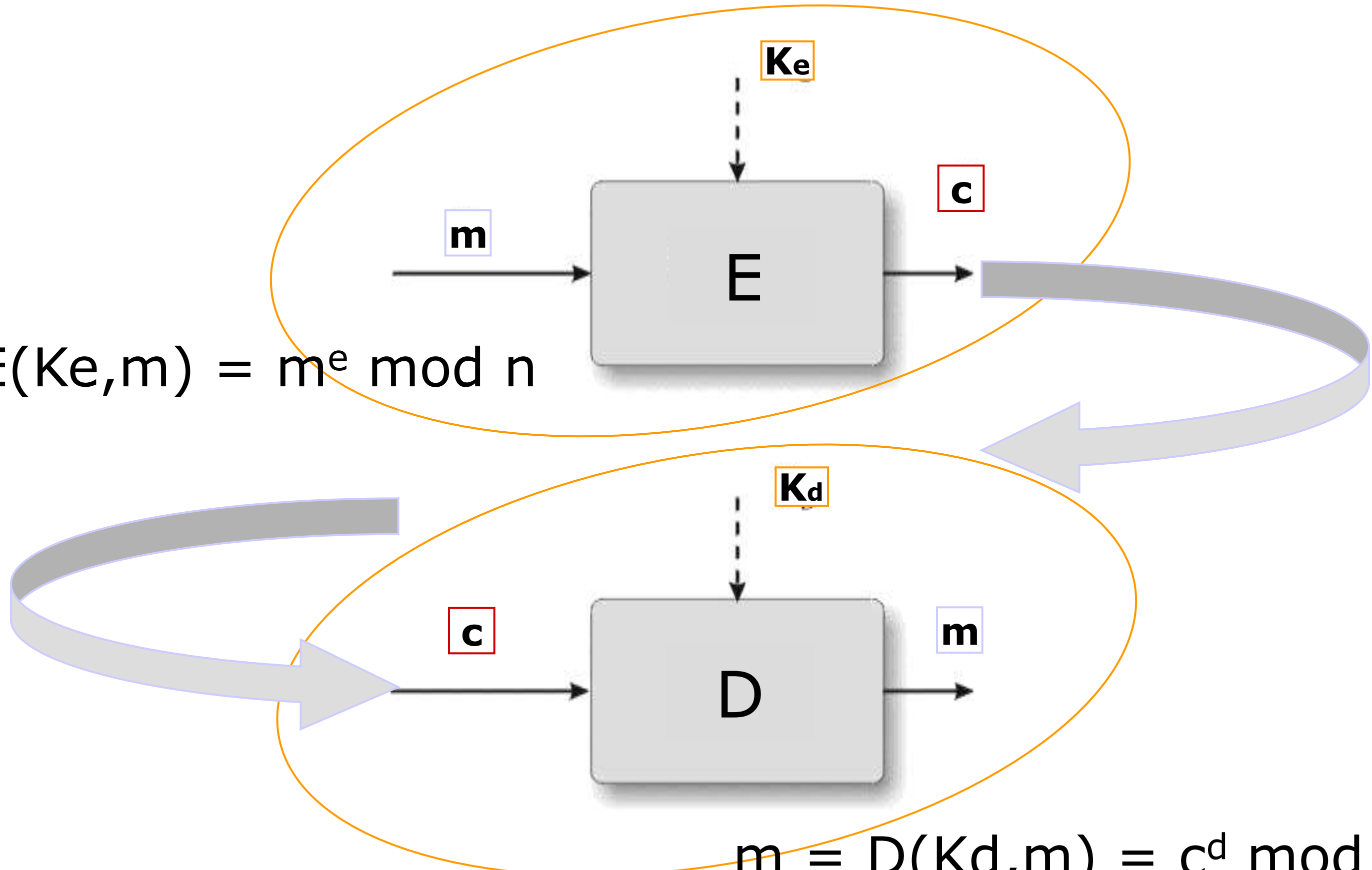
Rivest, Shamir, and Adleman (RSA)

- By Rivest, Shamir & Adleman of MIT in 1977
- Best known & widely used public-key scheme
- Based on exponentiation in a finite (Galois) field over integers modulo n
  - **n.b. exponentiation takes $O((\log n)^3)$ operations (easy)**
- uses large integers (eg. 1024 bits)
- security due to cost of factoring large numbers
  - **nb. factorization takes $O(e^{\log n \log \log n})$ operations (hard)**
- The key length is variable
  - **long keys for enhanced security, or a short keys for efficiency**
- The plaintext block size (the chunk to be encrypted) is also variable
  - **The plaintext block size must be smaller than the key length**
  - **The ciphertext block will be the length of the key**
- RSA is much slower to compute than popular secret key algorithms like DES, IDEA, and AES

RSA Algorithm

- First, you need to generate a public key and a corresponding private key:
  - **choose two large primes *p* and *q* (around 512 bits each or more)**
    - *p* and *q* will remain secret
  - **multiply them together (result is 1024 bits), and call the result *n***
    - it's practically impossible to factor numbers that large for obtaining *p* and *q*
  - **compute $\phi(n) = (p-1)(q-1)$**
  - **choose a number *e* that is relatively prime (that is, it does not share any common factors other than 1) to $\phi(n)$**
  - **find the number *d* that is the multiplicative inverse of *e* mod $\phi(n)$**
  - **your public key is $KU = K^+ = <e,n>$**
  - **your private key is $KR = K^- = <d,n>$ (or $<d,p,q>$)**
- To encrypt a message *m* (< *n*), someone can use your public key
  - **$c = m^e \bmod n$**
- Only you will be able to decrypt *c*, using your private key
  - **$m = c^d \bmod n$**

Textbook RSA



$c = E(Ke,m) = m^e \bmod n$

$m = D(Kd,m) = c^d \bmod n$

- **m** **plaintext**
- **c** **ciphertext**
- **Ke** **encription key (e.g. public key, KU or K^+)**
- **Kd** **decription key (e.g. private key, KR or K^-)**

RSA Key Setup

- Each user generates a public/private key pair by:
  - **selecting two large primes at random** p, q
  - **computing their system modulus** $n = p \cdot q$
    - note $\varnothing(n) = (p-1)(q-1)$
  - **selecting at random the encryption key** e
    - where $1 < e < \varnothing(n)$, $\gcd(e, \varnothing(n)) = 1$
  - **solve following equation to find decryption key** d
    - $e\ d = 1 \bmod \varnothing(n)$ and $0 \le d \le n$
- Publish their public encryption key: KU = {e,n}
- Keep secret private decryption key: KR = {d,n} or {d,p,q}

RSA Use

- To encrypt a message *m* the sender:
  - **obtains public key of recipient** $KU=<e,n>$
  - **computes:** $c=m^e \bmod n$, **where** $0 \le m < n$
- To decrypt the ciphertext *c* the owner:
  - **uses their private key** $KR=<d,n>$
  - **computes:** $m=c^d \bmod n$
- Note that the message *m* must be smaller than the modulus *n*
  - **it is a block cipher, where block size depends on the length of the modulus**
  - **if *m* is longer, CBC or other block cipher encryption modes can be used**
    - much slower than symmetric encryption

Why RSA Works

- Because of Euler's Theorem:
  - $a^{\varnothing(n)} \bmod n = 1$
    - where *gcd(a,n)=1*

  also:
  - $a^{k\varnothing(n)} \bmod n = 1^k = 1$

  and:
  - $a^{k\varnothing(n)+1} \bmod n = a$

- In RSA have:
  - **$n=p \cdot q$**
  - **$\varnothing(n)=(p-1)(q-1)$**
  - **carefully chosen *e* and *d* to be inverses mod *ø(n)***
    - hence $e \cdot d=1+k \cdot \phi(n)$ for some k
- Encryption:
  - **$c = E(K_e,m) = m^e \bmod n$**
- Decryption:
  - **$D(K_d,c) = c^d \bmod n = (m^e)^d \bmod n = m^{ed} \bmod n = m^{1+k\varnothing(n)} \bmod n = m$**

RSA Example

RSA setup

- select primes: p=17 & q=11
- compute n = pq =17×11=187
- compute ø(n)=(p–1)(q-1)=16×10=160
- select e : gcd(e,160)=1; choose e=7
- determine d: de=1 mod 160 and d < 160 Value is d=23 since 23×7=161= 160+1
- publish public key KU = K^+ = {7,187}
- keep secret private key KR = K^- = {23,187} = {23,17,11}

RSA Example (cont)

Textbook RSA encryption/decryption:

- given message `M = 88` (nb. `88<187`)
- encryption:
  $C = 88^{7} \bmod 187 = 11$
- decryption:
  $M = 11^{23} \bmod 187 = 88$

Textbook RSA is not secure

- Textbook RSA encryption:
  - **public key: (n,e) Encrypt: $c = m^e \pmod{n}$, with m<n**
  - **private key: (n,d) Decrypt: $m = c^d \pmod{n}$**
- If *n=pq* is large, the factorization of *n* is practically impossibly
- However, many attacks exist to Textbook RSA
  - **examples**
    - if *e* is small, and the message *m* is < $n^{1/e}$, then m^e is < *n* and computation of *c* doesn't involve any modular reduction
    - if *m* is small, knowing *c* and *{e,n}* a brute force search is possible
      - an improvement of the this attack also exist, that does not require a brute force search
    - common modulus attack - if several keys share the same modulus *n*, if the same *m* is encrypted with two keys $\{e_1,n\}$ and $\{e_2,n\}$ with $gcd(e_1, e_2)=1$, an adversary who sees c_1 and c_2 and knows the two public keys can recover *m*
      - $gcd(e_1,e_2)=1 \rightarrow ue_1+ve_2=1 \rightarrow c_1^u{}_*c_2^v = m^{e1u} * m^{e2v} = m^{ue1+ve2} = m$

Padded RSA

- $c = E(k_e,m) = (pad(m))^e \bmod n$
- $m = pad^{-1}(m')$, with $m' = c^d \bmod n$
- Different padding schemes exist, examples:
  - **PKCS#1 - v1.5**
  - **PKCS#1 - v2 OAEP (Optimal Asymmetric Encryption Padding)**

PKCS#1 - V1.5

- Encoded message EM = 0x00 || 0x02 || PS || 0x00 || M

  where:

  - **len(m) up to L-11 octets, where L is the octet length of the RSA modulus**
  - **PS is padding string consisting of pseudo-randomly generated nonzero octets**
    - len(PS) at least eight octets

PKCS#1 - V2 OAEP

- Given:
  - **MGF(seed,len)**
    - Mask Generation Function that generates *len* pseudo-random octects using a given *seed*
  - **H(x)**
    - hash function with length *h*
- Encoded message EM = maskedSeed || maskedDB
  - **with len(M) < len(EM) -2h-1**
  - **Where:**
    - P = a parameter string or label or null string
    - data block DB = H(P) || PS || 01 || M
      - with len(DB) = h + len(PS) + 1 +len(M) = len(EM) - h
    - seed = random octet string
      - with len(seed) = h
    - maskedDB = DB XOR MGF(seed, len(EM)-h)
      - with len(maskedDB) = len(EM)-h
    - maskedSeed = seed XOR MGF(maskedDB, h)
      - with len(maskedSeed) = h

RSA Security

- Three main approaches to attacking RSA:
  - **brute force key search**
    - brute force search on key space
      - infeasible given the key size
  - **cryptographic attacks**
    - try to find the private key by finding ø(n), by factoring modulus n and find p and q
      - infeasible given the size of n
    - other attacks in case *p,q,e,d* values are not selected properly
  - **timing attacks**
    - by measuring the time spent on running decryption

RSA Security (cont.)

- Cryptographic attacks if *p,q,e,d* are not selected properly:
  - **Modulus too small**
    - if the RSA key is too short, the modulus can be factored by just using brute force
  - **Low private exponent**
    - the smaller d is, the faster this operation goes
      - note: If the private exponent is small, the public exponent is necessarily large, so a public key with a large public exponent, is a good hint for attackers
  - **Low public exponent**
    - having a low public exponent makes the system vulnerable to certain attacks if used incorrectly
  - **Generator p and q close together**
    - if $p \approx q$, then $n \approx p^2$ and n can be efficiently factored using Fermat's factorization method

Progress in factorization (from Wikipedia)

RSA number	Decimal digits	Binary digits	Cash prize offered	Factored on	Factored by
RSA-100	100	330	US$1000	April 1, 1991	Arjen K. Lenstra
RSA-110	110	364	US$4429	April 14, 1992	Arjen K. Lenstra and M.S. Manasse
RSA-120	120	397	US$5898	July 9, 1993	T. Denny et al.
RSA-129	129	426	US$100	April 26, 1994	Arjen K. Lenstra et al.
RSA-130	130	430	US$14527	April 10, 1996	Arjen K. Lenstra et al.
RSA-140	140	463	US$17226	February 2, 1999	Herman te Riele et al.
RSA-150	150	496		April 16, 2004	Kazumaro Aoki et al.
RSA-155	155	512	US$9383	August 22, 1999	Herman te Riele et al.
RSA-160	160	530		April 1, 2003	Jens Franke et al., University of Bonn
RSA-170	170	563		December 29, 2009	D. Bonenberger and M. Krone
RSA-576	174	576	US$10000	December 3, 2003	Jens Franke et al., University of Bonn
RSA-180	180	596		May 8, 2010	S. A. Danilov and I. A. Popovyan, Moscow State University
RSA-190	190	629		November 8, 2010	A. Timofeev and I. A. Popovyan
RSA-640	193	640	US$20000	November 2, 2005	Jens Franke et al., University of Bonn
RSA-200	200	663		May 9, 2005	Jens Franke et al., University of Bonn
RSA-210	210	696		September 26, 2013	Ryan Propper
RSA-704	212	704	US$30000	July 2, 2012	Shi Bai, Emmanuel Thomé and Paul Zimmermann
RSA-220	220	729		May 13, 2016	S. Bai, P. Gaudry, A. Kruppa, E. Thomé and P. Zimmermann
RSA-230	230	762		August 15, 2018	Samuel S. Gross, Noblis, Inc.
RSA-232	232	768		February 17, 2020	N. L. Zamarashkin, D. A. Zheltkov and S. A. Matveev.
RSA-768	232	768	US$50000	December 12, 2009	Thorsten Kleinjung et al.
RSA-240	240	795		Dec 2, 2019	F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé and P.
RSA-250	250	829		Feb 28, 2020	F. Boudot, P. Gaudry, A. Guillevic, N. Heninger, E. Thomé and P.

2007
End of challenge

Using both symmetric and asymmetric cryptography

- Confidentiality can be provided through either symmetric or asymmetric encryption
- Requires that the two (or more) parties share:
  - **the secret key, in case of symmetric encryption, or**
  - **the public key of the sender, in case of asymmetric encryption**
- Usually symmetric encryption is preferred when
  - **a long message have to be encrypted**
  - **multiple messages have to be sent**
- In this case, if the two parties share only public keys, public key cryptography can be used for exchanging a symmetric (secret) key
  - **this secret key is sometimes referred as session key**

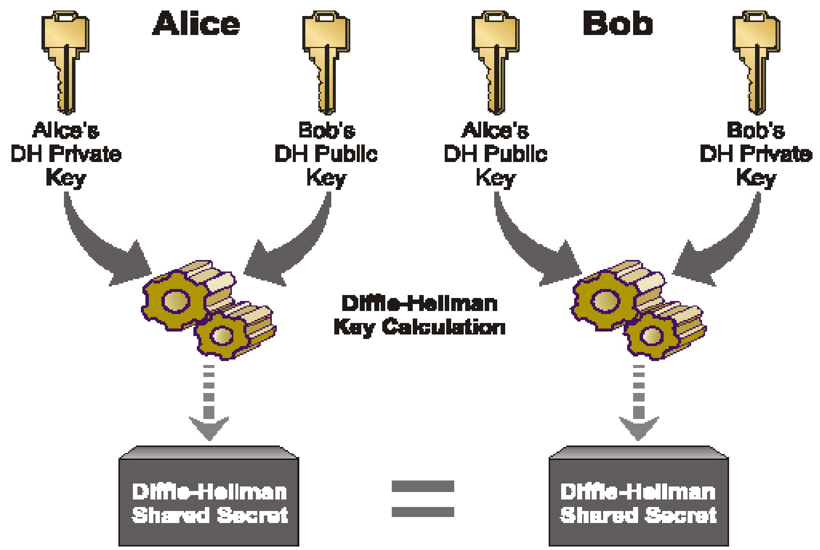
Using both symmetric and asymmetric cryptography (cont.)

- Example of encryption of a message m from A to B using symmetric cipher without having a pre-shared symmetric secret key, in one pass:
  - **A $\rightarrow$ B: E(Ks,m) , {Ks}K_B^+**
  - **where:**
    - E(K,x) : symmetric encryption (e.g. AES-CBC)
    - {x}K^+ : public key encryption (e.g. RSA)

- Other mechanisms are possible, involving more passes (exchanges) using a key establishment (key agreement) protocol
  - **e.g. using authenticated Diffie-Hellman exchange**

Diffie-Hellman (DH)

Diffie-Hellman



Diffie-Hellman

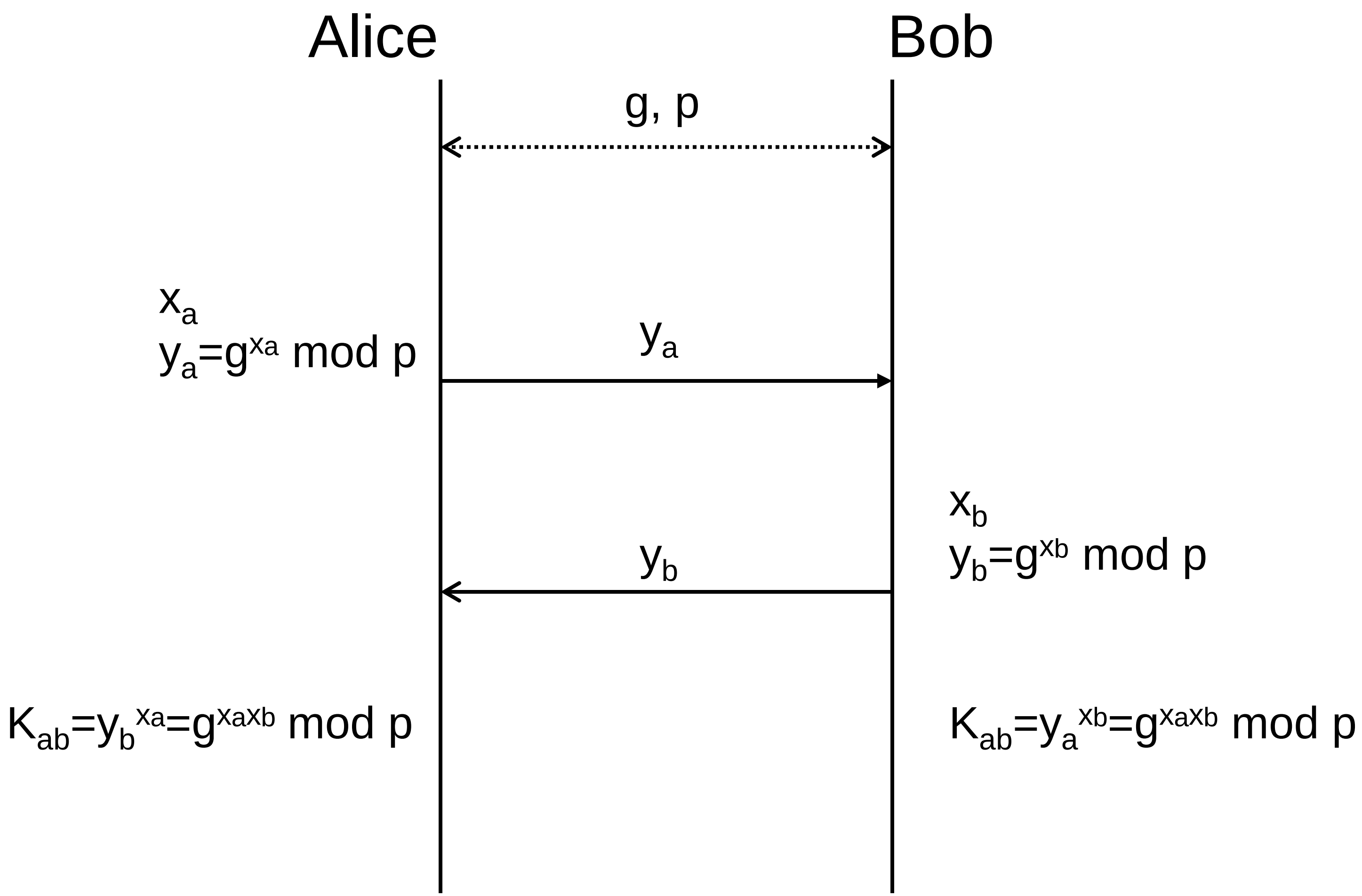
- First public-key type scheme proposed
- By Diffie & Hellman in 1976 along with the exposition of public key concepts
  - **now know that James Ellis (UK CESG) secretly proposed the concept in 1970**
    - predates RSA
  - **less general than RSA: it does neither encryption nor signature**
- Is a practical method for public exchange of a secret key
  - **allows two individuals to agree on a shared secret (key)**
  - **It is actually used for key establishment**
- Used in a number of commercial products

Diffie-Hellman Setup

Diffie-Hellman setup:

- all users agree on global parameters:
  - ***p*** **= a large prime integer or polynomial**
  - ***g*** **= a primitive root mod** ***p***
- each user (eg. A) generates their key
  - **chooses a secret key (number):** $x_A < p$
  - **compute their** public key**:** $y_A = g^{x_A} \bmod p$
- each user makes public that key y_A

Diffie-Hellman Key Exchange



Diffie-Hellman Key Exchange

Key exchange:

- Shared key K_{AB} for users A & B can be computed as:

  $$K_{AB} = g^{x_A x_B} \bmod p$$
  $$= y_B^{x_A} \bmod p \quad \text{(which A can compute)}$$
  $$= y_A^{x_B} \bmod p \quad \text{(which B can compute)}$$

- K_{AB} can be used as session key in secret-key encryption scheme between A and B

- Attacker must solve discrete log
  - **hard problem if *p* is chosen properly**

- Requires an integrity protected channel
  - **otherwise it is vulnerable to Man-In-The-Middle (MITM) attack**

Diffie-Hellman - Example

- users Alice & Bob who wish to swap keys:
- agree on prime $p=353$ and $g=3$
- select random secret keys:
  - **A chooses $x_A=97$, B chooses $x_B=233$**
- compute public keys:
  - **$y_A=3^{97} \bmod 353 = 40$ (Alice)**
  - **$y_B=3^{233} \bmod 353 = 248$ (Bob)**
- compute shared session key as:

$K_{AB}= y_B^{x_A} \bmod 353 = 248^{97} = 160$ (Alice)
$K_{AB}= y_A^{x_B} \bmod 353 = 40^{233} = 160$ (Bob)

Security uses of public key cryptography

- Transmitting over an insecure channel
  - **e.g. RSA**
    - each party has a <public key, private key> pair (Ku,Kr)
    - each party encrypts with the public key of the other party

encrypt m_A using Ku_B	$\longrightarrow$	**decrypt m_A using Kr_B**
decrypt m_B using Kr_A	$\longleftarrow$	**encrypt m_B using Ku_A**

- Secure storage on insecure media
  - **e.g. RSA**
    - encrypt with public key, decrypt with private key
    - useful when you can let third party to encrypt data
- Data authentication (Digital signature)
  - **e.g. DSA, RSA signature**
- Key establishment
  - **e.g. Diffie-Hellman**

Security uses of public key cryptography

- Peer Authentication (identification)
  - **Zero Knowledge Proof schemes**
    - prove that you know a secret without leaking any information
      - an entity A (prover) identifies itself by proving knowledge of a secret to any verifier B, without revealing any information about the secret, not known or computable by B prior to execution of the algorithm
  - **RSA**
    - authentication by proving the knowledge of the private key

encrypt r using Ku_B $\longrightarrow$ **decrypt to r using Kr_B**

$\longleftarrow$ **r**

- Note
  - **Public key cryptography has specific algorithm for specific function such as**
    - data encryption
    - MAC/digital signature
    - key establishment
    - peer authentication

Pros and cons of Public key cryptography

- Pros
  - **Every users have to keep only one secret (the private key)**
    - public keys of other users can verified through a trusted third party infrastructure (e.g. PKI)
  - **The total number of keys for *N* users is *2N***
    - instead, with symmetric cryptography n(n-1)/2 keys are needed
- Cons
  - **Slower**
    - known public-key cryptographic algorithms are orders of magnitude slower than the best known secret key cryptographic algorithms